

Stratified Bayesian Optimization

Saul Toscano-Palmerin, Peter I. Frazier

st684@cornell.edu, pf98@cornell.edu

School of Operations Research & Information Engineering

Cornell University

Ithaca, NY 14853

Abstract

We consider derivative-free black-box global optimization of expensive noisy functions, when most of the randomness in the objective is produced by a few influential scalar random inputs. We present a new Bayesian global optimization algorithm, called Stratified Bayesian Optimization (SBO), which uses this strong dependence to improve performance. Our algorithm is similar in spirit to stratification, a technique from simulation, which uses strong dependence on a categorical representation of the random input to reduce variance. We demonstrate in numerical experiments that SBO outperforms state-of-the-art Bayesian optimization benchmarks that do not leverage this dependence.

1 Introduction

We consider derivative-free black-box global optimization of expensive noisy functions,

$$\max_{x \in A \subset \mathbb{R}^n} \mathbb{E}[f(x, w, z)], \quad (1)$$

where the expectation is taken over $z \in \mathbb{R}^{d_1}$ and $w \in \mathbb{R}^{d_2}$, which have joint probability density p , A is a simple compact set (e.g., a hyperrectangle, or simplex), and we can directly observe only $f(x, w, z)$ at some collection of chosen or sampled x, w, z , and not its expectation, or the derivative of this expectation. We suppose that f has no special structural properties, e.g., concavity, or linearity, that we can exploit to solve this problem, making it a “black box.” We also suppose that evaluating f is costly or time-consuming, making these evaluations “expensive”, severely limiting the number of evaluations we may perform. This typically occurs because each evaluation requires running a complex PDE-based or discrete-event simulation, or requires training a machine learning algorithm on a large dataset. When f comes from a discrete-event simulation, this problem is also called “simulation optimization.”

Bayesian optimization is a popular class of techniques for solving this problem, originating with the seminal paper (Kushner, 1964), and enjoying early contributions from (Mockus *et al.*, 1978; Mockus, 1989). This class of techniques was popularized in the 1990s by the introduction in (Jones *et al.*, 1998) of the most well-known Bayesian optimization method, Efficient Global Optimization (EGO), relying on earlier ideas from (Mockus, 1989). Recently the machine learning community has devoted considerable attention to Bayesian optimization for its applications to tuning computationally intensive machine learning models, as in, e.g., (Snoek *et al.*, 2012). Textbooks and surveys on Bayesian optimization include (Forrester *et al.*, 2008; Brochu *et al.*, 2010).

Most work on Bayesian optimization assumes we can observe the objective function directly without noise, but a substantial number of papers, e.g. (Villemonteix *et al.*, 2009; Huang *et al.*, 2006; Scott *et al.*, 2011; Brochu *et al.*, 2010), do allow noise and thus consider (1). These methods all build a statistical model (usually

using Gaussian processes) of the function $x \mapsto G(x) := \mathbb{E}[f(x, w, z)]$ using noisy observations, and then use an acquisition criterion, typically expected improvement or probability of improvement (Brochu *et al.*, 2010), to decide where to sample next.

Existing work from Bayesian optimization for solving (1) relies on noisy evaluations in which w and z are drawn iid from their governing joint probability distribution p , and then $f(x, w, z)$ is observed. However, in many applications, we have the ability to choose not just x , but w as well, simulating the remaining components z conditioning on these values. (The choice of which random inputs to include in w and which in z was arbitrary in (1), but will be assumed below to accommodate this distinction.) This ability to simulate random inputs given the value of some of their values is widely used in stratified sampling to estimate expectations with better precision (Glasserman, 2003).

For example, in a queuing simulation (we give a detailed example in our numerical experiments), we can simulate the individual arrival times of customers z conditioning on the overall number of arrivals w . In a revenue management simulation, we can simulate individual purchase decisions z conditioned on the overall demand w . In an aerodynamic simulation, we can simulate fine-scale airflows z , conditioned on average wind speed w .

We thus rephrase problem (1) into the equivalent problem

$$\max_{x \in A \subset \mathbb{R}^n} \mathbb{E}[F(x, w)] \quad (2)$$

where $F(x, w) := \mathbb{E}[f(x, w, z) \mid w]$, and the problems are equivalent because $\mathbb{E}[F(x, w)] = \mathbb{E}[f(x, w, z)] = G(x)$.

This equivalent formulation suggests that standard approaches to Bayesian optimization are wasteful from a statistical point of view, as they do not use past observations w to learn $G(x) = \mathbb{E}[F(x, w)]$, treating w only as an unobservable source of noise. Instead, one can use Bayesian quadrature (O’Hagan, 1991), which builds a Gaussian process model of the function $F(x, w)$ using past observations of $x, w, f(x, w, z)$, and then uses the known relationship $G(x) = \int F(x, w)p(w)dw$ (where we assume $p(w) := \int p(w, z)dz$ is known in closed form) to imply a second Gaussian process model on $G(x)$.

In this paper, we leverage this ability and develop an algorithm, called stratified Bayesian optimization (SBO), which chooses not just the x at which to evaluate $f(x, w, z)$, but also the w . It chooses these using a one-step Bayes-optimal acquisition function based on a value-of-information (Howard, 1966) analysis. It then samples z from its conditional distribution given w , and uses the resulting observation within a Bayesian quadrature framework to update its Gaussian process posterior on both $(x, w) \mapsto F(x, w)$ and $x \mapsto \mathbb{E}[F(x, w)]$. By using more information, we make our statistical model more powerful, and provide better answers with fewer samples.

This approach is similar in spirit to stratified sampling (Glasserman, 2003), where our goal is to estimate $G(x) = \mathbb{E}[F(x, w)] = \mathbb{E}[f(x, w, z)]$ for a fixed x , and we choose which values of w at which to sample rather than sampling them from their marginal distribution, and then compensate for this choice via a known relationship between $F(x, w)$ and $G(x)$ to obtain lower variance estimates.

To choose x and w , SBO uses a decision-theoretic approach that models the utility resulting from solutions to the optimization problem (2). SBO finds the pair of values (x, w) at which to sample that maximizes the expected utility of the final solution, under the assumption, made for tractability, that we may take only one additional sample. Thus, our SBO algorithm is optimal in a decision-theoretic sense, in a one-step setting.

This one-step decision-theoretic approach follows the development of acquisition functions for other settings. In more traditional Bayesian optimization problems, the well-known expected improvement acquisition function (Mockus, 1989; Jones *et al.*, 1998) has this optimality property when observations are noise-free and the final solution must be taken from previously evaluated solutions (Frazier and Wang, 2015), and the knowledge-gradient (KG) method (Frazier *et al.*, 2009; Scott *et al.*, 2011) has this optimality property when the final solution is not restricted to be a previously evaluated solution, in both the noisy and noise-free setting.

Our approach also builds on, and significantly generalizes, the previous work (Xie *et al.*, 2012), which developed a similar method, but did not allow for the inclusion of unmodeled random inputs z , instead requiring

all inputs to be included and modeled statistically in w . This introduces a heavy computational and statistical burden when dealing with problems in which the combined dimension of w and z is large, which includes many complex stochastic models, significantly limiting its applicability.

This paper is organized as follows: §2 presents our statistical model. §3 presents the SBO algorithm. §4 describes the computation of the value of information and its derivative. §5 presents simulation experiments. §6 concludes.

2 Statistical Model

The SBO algorithm that we develop relies on a Gaussian process (GP) model of the underlying function F , which then implies (because integration is a linear function) a Gaussian process model over G . This statistical approach mirrors a standard Bayesian quadrature approach, but we summarize it here both to define notation used later, and because its application to Bayesian optimization is new.

We first place a Gaussian process prior distribution over the function F :

$$F(\cdot, \cdot) \sim GP(\mu_0(\cdot, \cdot), \Sigma_0(\cdot, \cdot, \cdot, \cdot)),$$

where μ_0 is a real-valued function taking arguments (x, w) , and Σ_0 is a positive semi-definite function taking arguments (x, w, x', w') . Common choices for μ_0 and Σ_0 from the Gaussian process regression literature (Rasmussen and Williams, 2006; Murphy, 2012), e.g., setting μ_0 to a constant and letting Σ_0 be the squared exponential or Matérn kernel, are appropriate here as well.

Our algorithm will take samples sequentially. At each time $n = 1, 2, \dots, N$, our algorithm will choose x_n and w_n based on previous observations. It will then take M samples of $f(x_n, w_n, z)$ and observe the average response. More precisely, it will sample $z_{n,m} \sim p(z | w_n)$ for $m = 1, \dots, M$ and observe $y_n = \frac{1}{M} \sum_{m=1}^M f(x_n, w_n, z_{n,m})$. The choice of M is an algorithm parameter, and should be chosen large enough that the central limit theorem may be applied, so that we may reasonably model the (conditional) distribution of y_n as normal. We will then have,

$$y_n | x_n, w_n \sim N(F(x_n, w_n), \sigma^2(x_n, w_n)/M),$$

where $\sigma^2(w, z) := \text{Var}(f(x, w, z) | w)$. We assume that this conditional variance is finite for all x and w . In updating the posterior, we also assume that we observe this value $\sigma^2(x_n, w_n)$, although in practice we estimate it using the empirical variance from our M samples.

Let $H_n = (y_{1:n}, w_{1:n}, x_{1:n})$ be the history observed by time n . Then, the posterior distribution on F at time n is

$$F(\cdot, \cdot) | H_n \sim GP(\mu_n(\cdot, \cdot), \Sigma_n(\cdot, \cdot, \cdot, \cdot)),$$

where μ_n and Σ_n can be computed using standard results from Gaussian process regression (Rasmussen and Williams, 2006). To support later analysis, expressions for μ_n and Σ_n are provided in the appendix.

We denote by \mathbb{E}_n , Cov_n , and Var_n the conditional expectation, conditional covariance, and conditional variance on F (and thus also on G , since G is specified by F) with respect to the Gaussian process posterior given H_n . By results from Bayesian quadrature (O'Hagan, 1991), which rely on the previously noted fact that $G(x) = \int F(x, w)p(w)dw$,

$$\mathbb{E}_n[G(x)] = \int \mu_n(x, w)p(w)dw := a_n(x), \quad (3)$$

$$\begin{aligned} \text{Cov}_n(G(x), G(x')) &= \\ &= \int \int \Sigma_n(x, w, x', w') p(w) p(w') dw dw'. \end{aligned} \quad (4)$$

Ignoring some technical details, the first line is derived using interchange of integral and expectation, as in $\mathbb{E}_n[G(x)] = \mathbb{E}_n[\int F(x, w)p(w)dw] = \int \mathbb{E}_n[F(x, w)p(w)]dw = \int \mu_n(x, w)p(w)dw$. The second line is derived similarly, though with more effort, by writing the covariance as an expectation, and interchanging expectation and integration.

3 Stratified Bayesian Optimization (SBO) Algorithm

Our SBO algorithm will choose points to evaluate using a value of information analysis (Howard, 1966), which maximizes the expected gain in the quality of the final solution to (1) that results from a sample.

To support this value of information analysis, we first consider the expected solution quality resulting for a particular set of samples. After n samples, if we were to choose the solution to (1) with the best expected quality with respect to the Bayesian posterior distribution on G , we would choose

$$x_n^* \in \arg \max_x \mathbb{E}_n [G(x)] = \arg \max_x a_n(x).$$

This is the Bayes-optimal solution when we are risk neutral. This solution has expected value (again, with respect to the posterior),

$$\mu_n^* := \max_x \mathbb{E}_n [G(x)] = \max_x a_n(x).$$

The improvement in expected solution quality that results from a sample at (x, w) at time n is

$$V_n(x, w) = \mathbb{E}_n [\mu_{n+1}^* - \mu_n^* \mid x_{n+1} = x, w_{n+1} = w]. \quad (5)$$

We refer to this quantity as the *value of information*, and if we have one evaluation remaining, then choosing to sample at the point with the largest value of information is optimal from a Bayesian decision-theoretic point of view. If we have more than one evaluation remaining, then it is not necessarily Bayes-optimal, but we argue that it remains a reasonable heuristic.

Thus, our Stratified Bayesian Optimization (SBO) algorithm is defined by

$$(x_{n+1}, w_{n+1}) \in \arg \max_{x, w} V_n(x, w). \quad (6)$$

Detailed computation of this value of information, and its gradient with respect to x and w , is discussed below in §4. We use this gradient to solve (6) using multi-start gradient ascent or multi-start sequential least squares programming (Kraft *et al.*, 1988).

The SBO algorithm is summarized in Algorithm 1. The complexity of the SBO algorithm is $O(LN^2 + N^4)$ if it is run during N iterations, and L is the number of points in the discretization of the domain of the points x , see §4.

Algorithm 1 SBO Algorithm

- 1: **First stage of samples** Evaluate F at n_0 points, chosen uniformly at random from A . Use maximum likelihood or maximum a posteriori estimation to fit the parameters of the GP prior on F , conditioned on these n_0 samples. Let μ_0 and Σ_0 be the mean function and covariance kernel of the resulting GP posterior on F .
 - 2: **Main stage of samples:**
 - 3: **for** $n = 1$ **to** N **do**
 - 4: Update our Gaussian process posterior on F using all samples from the first stage, and samples $x_{1:n}, w_{1:n}, y_{1:n}$. This allows computation of μ_n and Σ_n as described in the appendix, computation of a_n through (3), and computation of V_n and ∇V_n as described in §4.
 - 5: Solve $(x_{n+1}, w_{n+1}) \in \arg \max_{x, w} V_n(x, w)$ using multi-start sequential least squares programming or multi-start gradient ascent and the ability to compute ∇V_n . Let (x_{n+1}, w_{n+1}) be the resulting maximizer.
 - 6: Evaluate $y_{n+1} = \frac{1}{M} \sum_{m=1}^M f(x_{n+1}, w_{n+1}, z_{n+1,m})$ where $z_{n+1,m}$ are iid draws from $p(z \mid w_{n+1})$, and $p(z \mid w) = p(w, z)/p(w)$ is the conditional density of z given w .
 - 7: **end for**
 - 8: Return $x^* = \arg \max_x a_{N+1}(x)$.
-

Figure 1 illustrates how SBO works, showing one step in the algorithm applied to a simple analytic test problem

$$\max_{x \in [-\frac{1}{2}, \frac{1}{2}]} \mathbb{E}[f(x, w, z)] = \max_{x \in [-\frac{1}{2}, \frac{1}{2}]} \mathbb{E}[zx^2 + w] \quad (7)$$

where $w \sim N(0, 1)$ and $z \sim N(-1, 1)$. Direct computation shows $F(x, w) = -x^2 + w$ and $G(x) = -x^2$.

The figure shows the contours of $F(x, w)$, the mean of SBO's posterior on $F(x, w)$ in the first row, and the value of information and SBO's posterior on $G(x)$ in the second row, all after $n = 7$ samples.

SBO's value of information is small near where SBO has already sampled, because it has less uncertainty about $F(x, w)$ in this region. Its value of information is also smaller for w far away from 0 because they have smaller $p(w)$, and thus their $F(x, w)$ have less influence on $G(x)$. SBO's value of information is also small for extreme values of x , because its posterior on G suggests that these x are far from its maximum. SBO's value of information is thus largest for points that are far from previous samples, closer to $x = 0$, and closer to $w = 0$, and SBO samples next at the point with the largest value of information.

The figure's bottom row shows equivalent quantities for the KG method, which, like other Bayesian optimization methods, models $G(x)$ directly, ignoring valuable information from w , and computes a value of information as a function of x only (it believes that observing near $x = 0.1$ or $x = 1$ would be most useful), leaving the choice of w to chance. Furthermore, after $n = 7$ observations, SBO's use of w allows it to have a much more accurate estimate of $G(x)$, and the location of its maximum.

4 Computation of the Value of Information and Its Gradient

In this section we discuss computation of the value information (5) and its gradient, to support implementation of the SBO algorithm. Due to space considerations, we keep our descriptions brief, especially in §4.1 and §4.2, and detailed derivations may be found in the appendix. Table 1 summarizes notation used in this section.

Table 1: Table of Notation.

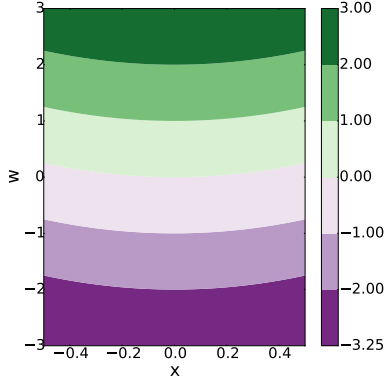
$G(x)$	\triangleq	$\mathbb{E}[f(x, w, z)]$
V_n	\triangleq	Value of Information at time n
$a_n(x)$	\triangleq	$\mathbb{E}_n[G(x)]$
H_n	\triangleq	History observed by time n
Σ_0	\triangleq	Kernel of the Gaussian process prior distribution over the function F
$B(x, i)$	\triangleq	$\int \Sigma_0(x, w, x_i, w_i) p(w) dw$, for $i = 1, \dots, n+1$
γ	\triangleq	$\begin{bmatrix} \Sigma_0(x_{n+1}, w_{n+1}, x_1, w_1) \\ \vdots \\ \Sigma_0(x_{n+1}, w_{n+1}, x_n, w_n) \end{bmatrix}$
A_n	\triangleq	$(\Sigma_0(x_i, w_i, x_j, w_j))_{i,j=1}^n + \text{diag}((\sigma^2(x_i, w_i))_{i=1}^n)$

4.1 Computation of the Value of Information

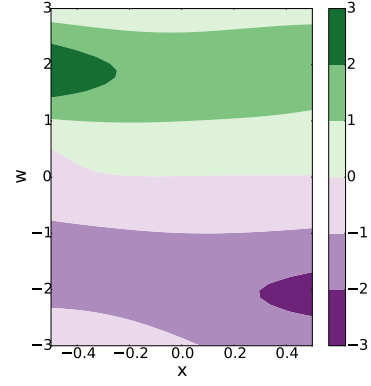
We first rewrite the value of information (5) as

$$V_n(x_{n+1}, w_{n+1}) = \mathbb{E}_n[\max_{x' \in A} a_{n+1}(x') \mid x_{n+1}, w_{n+1}] - \max_{x' \in A} a_n(x'). \quad (8)$$

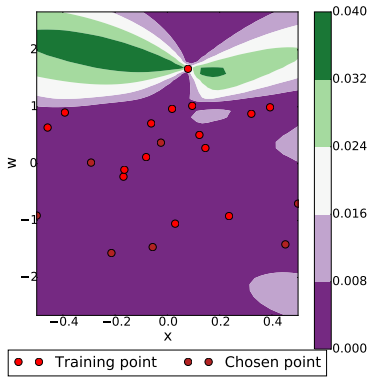
To calculate this expectation, we must find the joint distribution of $a_{n+1}(x)$ across all x conditioned on (x_{n+1}, w_{n+1}) and H_n for any x . This is provided by the following lemma.



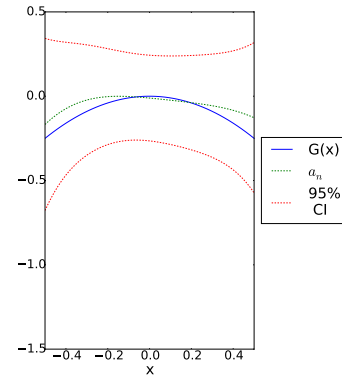
(a) The contours of $F(x, w)$. The objective $G(x)$ is $E[F(x, w)|x]$.



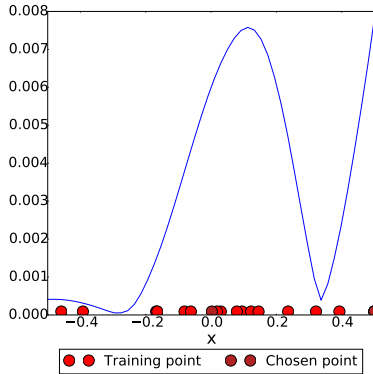
(b) SBO: The contours of SBO's estimate $\mu_n(x, w)$ of $F(x, w)$, at $n = 7$.



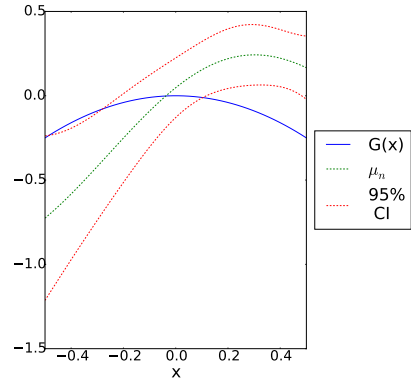
(c) SBO: The contours of the value of information $V_n(x, w)$ under SBO at $n = 7$. SBO's value of information depends on both x and w .



(d) SBO: The objective $G(x)$, and SBO's estimate $a_n(x)$ and 95% credible interval, at $n = 7$.



(e) KG: The value of information under KG at $n = 7$. KG's value of information depends only on x .



(f) KG: The objective $G(x)$, and KG's estimate $a_n(x)$ and 95% credible interval, at $n = 7$.

Figure 1: Illustration of the SBO algorithm on an analytic test problem. SBO models $F(x, w)$ while benchmark methods (KG and EI) model $G(x)$. **(First row)** The contours of $F(x, w)$ (left) and of SBO's estimate of F (right). **(Second row)** Left shows the contours of SBO's value of information, which depends on both x and w , and which SBO uses to choose the pair (x, w) to sample next. Right shows SBO's estimates of $G(x)$, which is based on the estimate of $F(x, w)$ in the first row. **(Third row)** Left shows KG's value of information, which depends only on x , and which KG uses to choose the point x to sample next. Right shows KG's estimates of $G(x)$. This estimate is of lower quality than SBO's estimate above, because it does not use the observed values of w .

Lemma 1. *There exists a standard normal random variable Z_{n+1} such that, for all x ,*

$$a_{n+1}(x) = a_n(x) + \tilde{\sigma}_n(x, x_{n+1}, w_{n+1})Z_{n+1}.$$

where

$$\begin{aligned} \tilde{\sigma}_n^2(x, x_{n+1}, w_{n+1}) &:= \text{Var}_n[G(x)] \\ &\quad - \mathbb{E}_n[\text{Var}_{n+1}[G(x)] \mid x_{n+1}, w_{n+1}]. \end{aligned}$$

To compute the value of information, we then discretize the feasible set A , over which we take the maximum in (8), into $L < \infty$ points. We let A' denote this discrete set of points, so $A' \subseteq A$ and $|A'| = L$. For example, if A is a hyperrectangle, then we may discretize it using a uniform mesh.

Then, we approximate (8) by

$$\begin{aligned} V_n(x_{n+1}, w_{n+1}) &= \mathbb{E}_n[\max_{x \in A} a_n(x) + \tilde{\sigma}(x, x_{n+1}, w_{n+1})Z_{n+1}] \\ &\quad - \max_{x \in A} a_n(x) \\ &\approx \mathbb{E}_n[\max_{x \in A'} a_n(x) + \tilde{\sigma}(x, x_{n+1}, w_{n+1})Z_{n+1}] \\ &\quad - \max_{x \in A'} a_n(x) \\ &= h(a_n(A'), \tilde{\sigma}_n(A', x_{n+1}, w_{n+1})), \end{aligned}$$

where $a_n(A') = (a_n(x_i))_{i=1}^L$, $\tilde{\sigma}_n(x, w) = (\tilde{\sigma}_n(x_i, x, w))_{i=1}^L$, and $h: \mathbb{R}^L \times \mathbb{R}^L \rightarrow \mathbb{R}$ is a function defined by $h(a, b) = \mathbb{E}[\max_i a_i + b_i Z] - \max_i a_i$, where a and b are any deterministic vectors, and Z is a one-dimensional standard normal random variable. By convenience, we will denote $a_n(x_i)$ by e_i and $\tilde{\sigma}_n(x_i, x, w)$ by f_i for each i in $\{1, \dots, L\}$. If $A = A'$, which is possible if A is a finite set, then the approximation in the second line above is exact.

In (Frazier *et al.*, 2009), it is also shown how to compute h . Using the Algorithm 1 in that paper, we can get a subset of indexes $\{j_1, \dots, j_\ell\}$ from $\{1, \dots, L\}$, such that

$$\begin{aligned} V_n(x_{n+1}, w_{n+1}) &= h(a_n(A'), \tilde{\sigma}_n(A', x_{n+1}, w_{n+1})) \\ &= \sum_{i=1}^{\ell-1} (f_{j_{i+1}} - f_{j_i}) f(-|c_i|) \end{aligned}$$

where

$$\begin{aligned} f(z) &:= \varphi(z) + z\Phi(z), \\ c_i &:= \frac{e_{j_{i+1}} - e_{j_i}}{f_{j_{i+1}} - f_{j_i}}, i = 1, \dots, \ell - 1, \end{aligned}$$

and φ, Φ are the standard normal cdf and pdf, respectively. This shows how to compute the Value of Information V_n .

4.2 Computation of the Gradient of the Value of Information

We show how to compute the gradient of the Value of Information V_n in this section. Observe that if $\ell = 1$, $V_n(x, w) = 0$ and so $\nabla V_n(x, w^{(1)}) = 0$. On the other hand, if $\ell > 1$, one can show via direct computation that

$$\nabla V_n(x, w) = \sum_{i=1}^{\ell-1} (-\nabla f_{j_{i+1}} + \nabla f_{j_i}) \varphi(|c_i|).$$

Consequently, we only need to compute ∇f_{j_i} for each i in $\{1, \dots, \ell\}$. Another direct computation shows that

$$\nabla \tilde{\sigma}_n(x, x_{n+1}, w_{n+1}) = \beta_1 \beta_3 - \frac{1}{2} \beta_1^3 \beta_2 [\beta_5 - \beta_4]$$

where

$$\begin{aligned} \beta_1 &= [\Sigma_0(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1}) - \gamma^T A_n^{-1} \gamma]^{-1/2}, \\ \beta_2 &= B(x, n+1) - [B(x, 1) \cdots B(x, n)] A_n^{-1} \gamma, \\ \beta_3 &= \left(\nabla B(x, n+1) - \nabla(\gamma^T) A_n^{-1} \begin{bmatrix} B(x, 1) \\ \vdots \\ B(x, n) \end{bmatrix} \right), \\ \beta_4 &= 2 \nabla(\gamma^T) A_n^{-1} \gamma, \\ \beta_5 &= \nabla \Sigma_0(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1}). \end{aligned}$$

4.3 Formulas for $\tilde{\sigma}_n(x, x_{n+1}, w_{n+1})$ and $a_n(x)$

Here, we give expressions for a_n to compute the parameters of the posterior distribution of a_{n+1} . First, a_n can be computed using the following formula,

$$\begin{aligned} a_n(x) &= \mathbb{E}[\mu_n(x, w)] \\ &= \mathbb{E}[\mu_0(x, w)] \\ &\quad + [B(x, 1) \cdots B(x, n)] A_n^{-1} \begin{pmatrix} y_1 - \mu_0(x_1, w_1) \\ \vdots \\ y_n - \mu_0(x_n, w_n) \end{pmatrix}. \end{aligned}$$

In some cases it is possible to get a closed-form formula for B , e.g. if w follows a normal distribution, the components of w are independent and we use the squared exponential kernel.

Finally, a direct computation detailed in the appendix shows that the formula for $\tilde{\sigma}_n^2(x, x_{n+1}, w_{n+1})$ is

$$\left[\frac{(B(x, n+1) - [B(x, 1) \cdots B(x, n)] A_n^{-1} \gamma)}{\sqrt{(\Sigma_0(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1}) - \gamma^T A_n^{-1} \gamma)}} \right]^2.$$

5 Numerical Experiments

We now present simulation experiments illustrating how the SBO algorithm can be applied in practice, and comparing its performance against some baseline Bayesian optimization algorithms. We compare on a test problem with a simple analytic form (§5.1), on a realistic problem arising in the design of the New York City's Citi Bike system (§5.2), and on a wide variety of problems simulated from Gaussian process priors (§5.3) designed to provide insight into what problem characteristics allow SBO to provide substantial benefit.

We consider two baseline Bayesian optimization algorithms. We use the Knowledge-Gradient policy of (Frazier *et al.*, 2009) and Expected Improvement criterion (Jones *et al.*, 1998), which both place the Gaussian process prior directly on $G(x)$, and use a standard sampling procedure, in which w and z are drawn from their joint distribution, and $f(x, w, z)$ is observed. Knowledge-Gradient policy is equivalent to SBO if all components of w are moved into z . Thus, comparing against KG quantifies the benefit of SBO's core contribution, while holding constant standard aspects of the Bayesian optimization approach.

We also solved the problems from (§5.1) and (§5.2) with Probability of Improvement (PI) (Brochu *et al.*, 2010), but we did not include its results in our graphs because both KG and EI outperformed PI. Moreover, according to Brochu (Brochu *et al.*, 2010), "EI's acquisition function is more satisfying than PI's acquisition function".

When implementing the SBO algorithm, we use the squared exponential kernel, which is defined as

$$\Sigma_0(x, w, x', w') = \sigma_0^2 \exp \left(- \sum_{k=1}^n \alpha_1^{(k)} [x_k - x'_k]^2 - \sum_{k=1}^{d_1} \alpha_2^{(k)} [w_k - w'_k(1)]^2 \right),$$

where σ_0^2 is the common prior variance and $\alpha_1^{(1)}, \dots, \alpha_1^{(n)}, \alpha_2^{(1)}, \dots, \alpha_2^{(d_1)} \in \mathbb{R}_+$ are length scales. These values, σ^2 and the mean μ_0 are calculated using maximum likelihood estimation following the first stage of samples.

5.1 An Analytic Test Problem

In our first example, we consider the problem (7) stated in §3. Figure 2 compares the performance of SBO, KG and EI on this problem, plotting the number of samples beyond the first stage on the x axis, and the average true quality of the solutions provided, $G(\arg\max_x \mathbb{E}_n[G(x)])$, averaging over 3000 independent runs of the three algorithms.

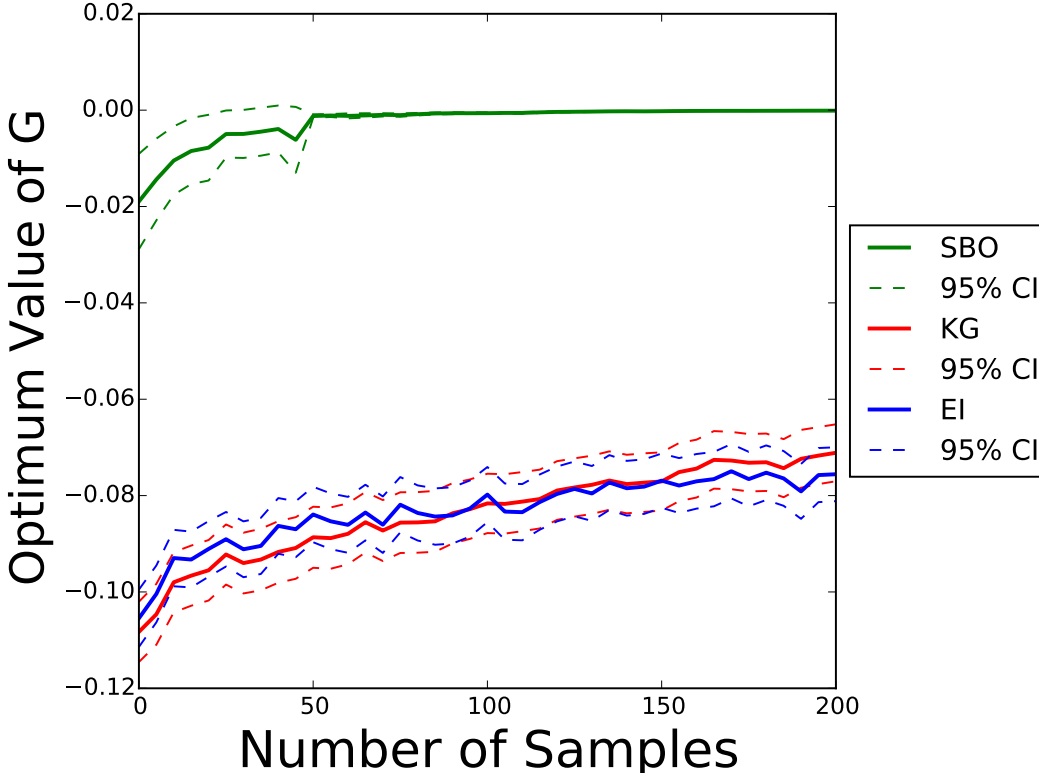


Figure 2: Performance comparison between SBO and two Bayesian optimization benchmark, the KG and EI methods, on the analytic test problem (7) from §3. SBO performs significantly better than two benchmarks: knowledge-gradient (KG) and expected improvement (EI).

We see that SBO substantially outperforms both benchmark methods. This is possible because SBO reduces the noise in its observations by conditioning on w , allowing it to more swiftly localize the objective's maximum.

5.2 New York City’s Citi Bike System

We now consider a more realistic problem, using a queuing simulation based on New York City’s Citi Bike system, in which system users may remove an available bike from a station at one location within the city, and ride it to a station with an available dock in some other location within the city. The optimization problem that we consider is the allocation of a constrained number of bikes (6000) to available docks within the city at the start of rush hour, so as to minimize, in simulation, the expected number of potential trips in which the rider could not find an available bike at their preferred origination station, or could not find an available dock at their preferred destination station. We call such trips “negatively affected trips.”

We simulated in Python the demand of bike trips of a New York City’s Bike System on any day from January 1st to December 31st between 7:00am and 11:00am. We used 329 actual bike stations, locations, and numbers of docks from the Citi Bike system, and estimated demand and average time for trips for every day in a year using publicly available data of the year 2014 from Citi Bike’s website (Citi, 2015).

We simulate the demand for trips between each pair of bike stations on a day using an independent Poisson process, and trip times between pairs of stations follows an exponential distribution. If a potential trip’s origination station has no available bikes, then that trip does not occur, and we increment our count of negatively affected trips. If a trip does occur, and its preferred destination station does not have an available dock, then we also increment our count of negatively affected trips, and the bike is returned to the closest bike station with available docks.

We divided the bike stations in 4 groups using k-nearest neighbors, and let x be the number of bikes in each group at 7:00 AM. We suppose that bikes are allocated uniformly among stations within a single group. The random variable w is the total demand of bike trips during the period of our simulation. The random vector z contains all other random quantities within our simulation.

Table 2 provides a concrete mapping of SBO’s abstractions onto the CitiBike example.

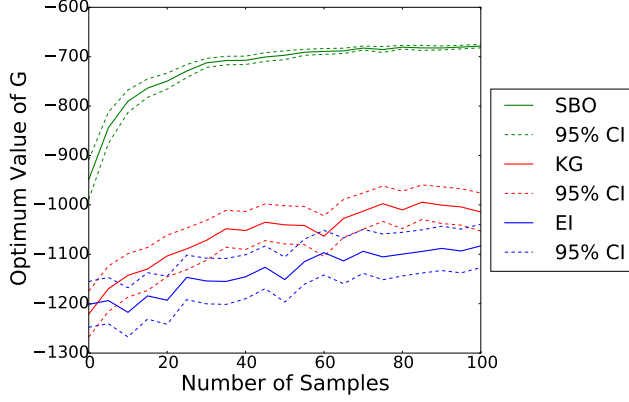
Table 2: Table of Notation for the Citibike Problem.

$x \in \mathbb{R}^4$	\triangleq	deterministic vector that represents the number of bikes in each group of bike stations at 7:00 AM.
$w \in \mathbb{N}$	\triangleq	Poisson random variable that represents the total demand of bike trips between 7:00am to 11:00am.
z	\triangleq	random vector that consists of: i) day of the year where the simulation occurs, ii) $\binom{329}{2}$ -dimensional Poisson random vector that represents the total demand between each pair of bike stations, iii) exponential random vector that represents the time duration of each bike trip.
$-f(x, w, z)$	\triangleq	negatively affected trips between 7:00am to 11:00am.
$G(x)$	\triangleq	$\mathbb{E}[f(x, w, z)]$.

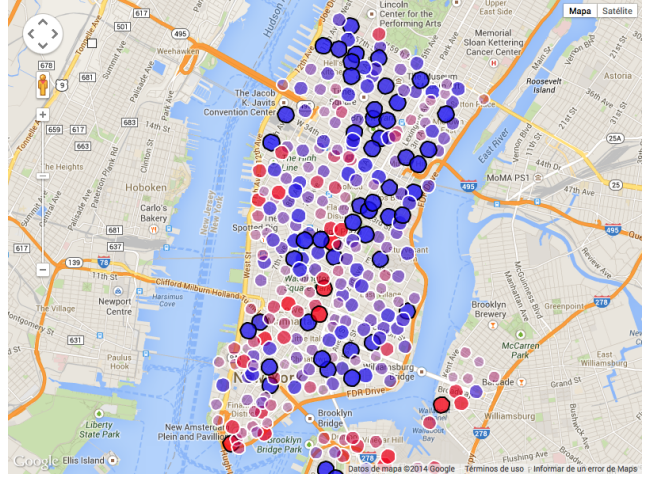
Figure 3a compares the performance of SBO, KG and EI, plotting the number of samples beyond the first stage on the x axis, and the average true quality of the solutions provided, $G(\arg\max_x \mathbb{E}_n[G(x)])$, averaging over 300 independent runs of the three algorithms. We see that SBO was able to quickly find an allocation of bikes to groups that attains a small expected number of negatively affected trips.

5.3 Problems Simulated from Gaussian Process Priors

We now compare the performance of SBO against a benchmark Bayesian optimization algorithm on synthetic problems drawn at random from Gaussian process priors. We use the KG algorithm as our benchmark, as it performed as well or better than the other benchmark algorithms (EI and Probability of Improvement) on the



(a) Performance comparison between SBO and two Bayesian optimization benchmark, the KG and EI methods, on the Citi Bike Problem from §5.2



(b) Location of bike stations (circles) in New York City, where size and color represent the ratio of available bikes to available docks.

Figure 3: Performance results for the Citi Bike problem (plot 1), and a screenshot from our simulation of the Citi Bike problem (plot b).

test problems in §5.1 and §5.2. In these experiments, SBO outperforms the benchmark on most problems, in some cases offering an improvement of almost 1000%. On those few problems in which SBO underperforms the benchmark, it underperforms by a much smaller margin of less than 50%.

Our experiments also provide insight into how SBO should be applied in practice. They show that the most important factor in determining SBO’s performance over benchmarks is the speed with which the conditional expectation $F(x, w) = \mathbb{E}[f(x, w, z) | w]$ varies with w . SBO provides the most value when this variation is large enough to influence performance, and small enough to allow $F(x, w)$ to be modeled with a Gaussian process. Thus, users of SBO should choose a w that plays a big role in overall performance, and whose influence on performance is smooth enough to support predictive modeling.

We now construct these problems in detail. Let $f(x, w, z) = h(x, w) + g(z)$ on $[0, 1]^2 \times \mathbb{R}$, where:

- $g(z)$ is drawn, for each z , independently from a normal distribution with mean 0 and variance α_d (we could have set g to be an Ornstein-Uhlenbeck process with large volatility, and obtained an essentially identical result).
- h is drawn from a Gaussian Process with mean 0 and Gaussian covariance function $\Sigma((x, w), (x', w')) = \alpha_h \exp\left(-\beta \|(x, w) - (x', w')\|_2^2\right)$.
- w is drawn uniformly from $\{0, 1/49, 2/49, \dots, 1\}$ and z is drawn uniformly from $[0, 1]$.

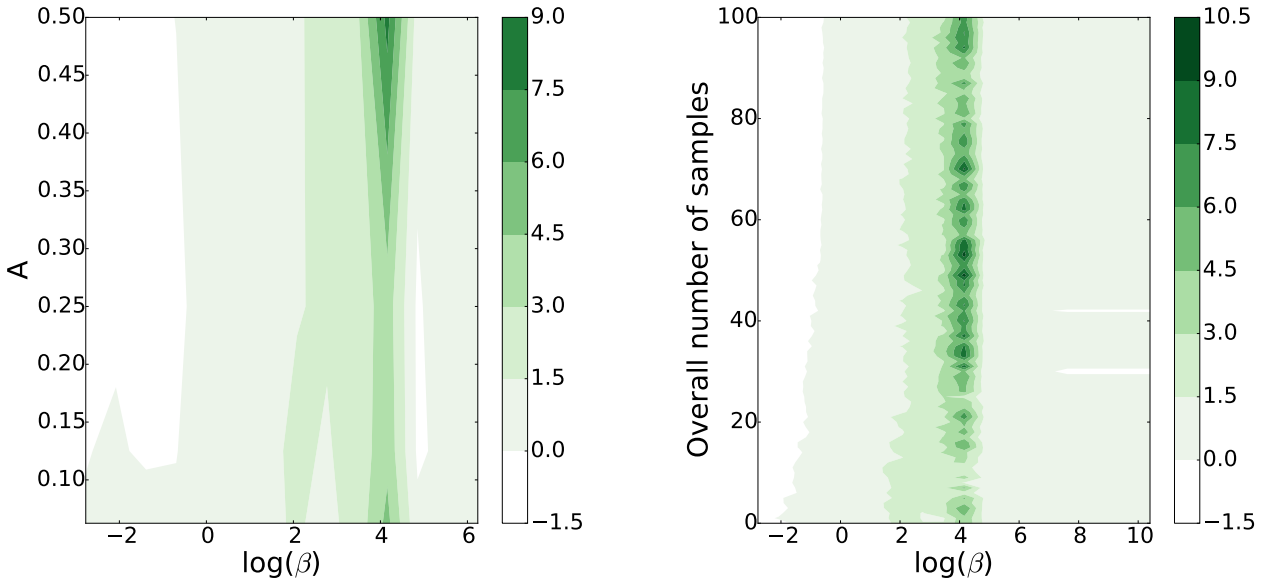
We thus have a class of problems parameterized by α_h , α_d , β , the number of samples per iteration n , and an outcome measure determined by the overall number of samples. To reduce the dimensionality of the search space, we first set the number of samples per iteration, n , to 1. (We also performed experiments with other n , not described here, and found the same qualitative behavior described below.)

We reparameterize the dependence on α_h and α_d in a more interpretable way. We first set $\text{Var}[f(x, w, z) | w, z] = \alpha_h + \alpha_d$ to 1, as multiplying both α_h and α_d by a scalar simply scales the problem. Then, the variance reduction ratio $\text{Var}[f(x, w, z) | f, w] / \text{Var}[f(x, w, z) | f]$ achieved by SBO in conditioning on w is approximately $\alpha_h / (\alpha_d + \alpha_h)$, with this estimate becoming exact as β grows large and the values of $h(x, w)$ become uncorrelated across w . We define $A = \alpha_h / (\alpha_d + \alpha_h)$ equal to this approximate variance reduction ratio.

Thus, our problems are parameterized by the approximate variance reduction ratio A , the overall number of samples, and by β , which measures the speed with which the conditional expectation $\mathbb{E}[f(x, w, z) | w]$ varies with w .

Given this parameterization, we sampled problems from Gaussian process priors using all combinations of $A \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$ and $\beta \in \{2^{-4}, 2^{-3}, \dots, 2^9, 2^{10}\}$. We also performed additional simulations at $A = \frac{1}{2}$ for $\beta \in \{2^{11}, \dots, 2^{15}\}$.

Figure 4 shows Monte Carlo estimates of the normalized performance difference between SBO and KG for these problems, as a function of $\log(\beta)$ (log is the natural logarithm), A , and the overall number of samples. The normalized performance difference is estimated for each set of problem parameters by taking a randomly sampled problem generated using those problem parameters, discretizing the domain into 2500 points, running each algorithm independently 500 times on that problem, and averaging $(G(x_{\text{SBO}}^*) - G(x_{\text{KG}}^*)) / |G(x_{\text{KG}}^*)|$ across these 500 samples, where x_{SBO}^* is the final solution calculated by SBO, and similarly for x_{KG}^* .



(a) Normalized performance difference as a function of β and A , when the overall number of samples is 50.

(b) Normalized performance difference as a function of β and the overall number of iterations, when $A = 1/2$.

Figure 4: Normalized performance difference between SBO and KG in problems simulated from a Gaussian process, as a function of β , which measures how quickly $\mathbb{E}[f(x, w, z) | w]$ varies with w , the approximate variance reduction ratio A , and the overall number of samples. SBO outperforms KG over most of the parameter space, and is approximately 10 times better when β is near $\exp(4)$.

We see that the normalized performance difference is robust to A and the overall number of samples, but is strongly influenced by β . We see that SBO is always better than KG whenever $\beta \geq 1$. Moreover, it is substantially better than KG when $\log(\beta) \in (3, 5)$, with SBO outperforming KG by as much as a factor of 10. For larger β , SBO remains better than KG, but by a smaller margin. This unimodal dependence of the normalized performance difference on β can be understood as follows: SBO provides value by modeling the dependence of $F(x, w)$ on w . Modeling this dependence is most useful when β takes moderate values because it is here where observations of $F(x, w)$ at one value of w are most useful in predicting the value of $F(x, w)$ at other values of w . When F varies very quickly with w (large β), it is more difficult to generalize, and when F varies very slowly with w (β close to 0), then modeling dependence on w is comparable with modeling F as constant.

6 Conclusion

We have presented a new algorithm called SBO for simulation optimization of noisy derivative-free expensive functions. This algorithm can be used with high dimensional random vectors, and it outperforms the classical Bayesian approach to optimize functions in the examples presented. Our algorithm can be 10 times better than the classical Bayesian approach, which is a substantial improvement over the standard approach.

Appendix

Statistical Model

In this section we compute the parameters of the posterior distribution of F and G .

Parameters of the posterior distribution of F

In this section we are going to calculate the posterior distribution of $F(\cdot, \cdot)$ given that we have placed a Gaussian process (GP) prior distribution over the function F :

$$F(\cdot, \cdot) \sim GP(\mu_0(\cdot, \cdot), \Sigma_0(\cdot, \cdot, \cdot, \cdot))$$

where

$$\begin{aligned} \mu_0 : (x, w) &\rightarrow \mathbb{R}, \\ \Sigma_0 : (x, w, x', w') &\rightarrow \mathbb{R}, \end{aligned}$$

and Σ_0 is a positive semi-definite function. We choose Σ_0 such that closer arguments are more likely to correspond to similar values, i.e. $\Sigma_0(x, w, x', w')$ is a decreasing function of the distance between (x, w) and (x', w') . Specifically, we can use the squared exponential covariance function:

$$\Sigma_0(x, w^{(1)}, x', w'^{(1)}) = \sigma_0^2 \exp \left(- \sum_{k=1}^n \alpha_{1,k} [x_k - x'_k]^2 - \sum_{k=1}^{d_1} \alpha_{2,k} [w_k - w'_k]^2 \right)$$

where σ_0^2 is the common prior variance, and $\alpha_{1,1}, \dots, \alpha_{1,n}, \alpha_{2,1}, \dots, \alpha_{2,d_1} \in \mathbb{R}_+$ are the length scales. These values are calculated using likelihood estimation from the observations of F .

First, observe that standard results from Gaussian process regression provide the following expressions for μ_n and Σ_n (the parameters of the posterior distribution of F),

$$\begin{aligned} \mu_n(x, w) &= \mu_0(x, w) \\ &\quad + [\Sigma_0(x, w, x_1, w_1) \cdots \Sigma_0(x, w, x_n, w_n)] A_n^{-1} \\ &\quad \times \begin{pmatrix} y_1 - \mu_0(x_1, w_1) \\ \vdots \\ y_n - \mu_0(x_n, w_n) \end{pmatrix} \\ \Sigma_n(x, w, x', w') &= \Sigma_0(x, w, x', w') \\ &\quad - [\Sigma_0(x, w, x_1, w_1) \cdots \Sigma_0(x, w, x_n, w_n)] A_n^{-1} \begin{pmatrix} \Sigma_0(x', w', x_1, w_1) \\ \vdots \\ \Sigma_0(x', w', x_n, w_n) \end{pmatrix} \end{aligned}$$

where

$$A_n = \begin{bmatrix} \Sigma_0(x_1, w_1, x_1, w_1) & \cdots & \Sigma_0(x_1, w_1, x_n, w_n) \\ \vdots & \ddots & \vdots \\ \Sigma_0(x_n, w_n, x_1, w_1) & \cdots & \Sigma_0(x_n, w_n, x_n, w_n) \end{bmatrix} + \text{diag}(\sigma^2(x_1, w_1), \dots, \sigma^2(x_n, w_n)),$$

and $\sigma^2(x, w) = \text{Var}(f(x, w, z) | w)$.

Parameters of the posterior distribution of G

In this section, we compute the parameters of the posterior distribution of G , $\tilde{\sigma}_n(x, x_{n+1}, w_{n+1})$ and $a_n(x)$. We give close formulas for these parameters when we use the squared exponential kernel, and w follows a normal distribution ($w_i \sim N(\mu_i, \sigma_i^2)$) and its components are independent.

We first compute $\tilde{\sigma}_n(x, x_{n+1}, w_{n+1})$,

$$\begin{aligned} & \tilde{\sigma}_n^2(x, x_{n+1}, w_{n+1}) \\ &= \text{Var}_n[G(x)] - \mathbb{E}_n[\text{Var}_{n+1}[G(x)] | x_{n+1}, w_{n+1}] \\ &= \text{Var}_n[G(x) | x_{n+1}, w_{n+1}] - \text{Var}_{n+1}[G(x) | x_{n+1}, w_{n+1}] \\ &= \int \int \Sigma_n(x, w, x, w') p(w) p(w') dw dw' \\ &\quad - \int \int \Sigma_{n+1}(x, w, x, w') p(w) p(w') dw^{(1)} dw'^{(1)} \\ &= \int \int \Sigma_n(x, w, x_{n+1}, w_{n+1}) \frac{\Sigma_n(x, w', x_{n+1}, w_{n+1})}{\Sigma_n(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1})} p(w) p(w') dw dw' \\ &= \left[\frac{\int \Sigma_n(x, w, x_{n+1}, w_{n+1})}{\sqrt{\Sigma_n(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1})}} p(w) dw \right]^2 \\ &= \left[\frac{\int \Sigma_n(x, w, x_{n+1}, w_{n+1})}{\sqrt{\Sigma_n(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1})}} p(w) dw \right]^2 \\ &= \left[\frac{(B(x, n+1) - [B(x, 1) \cdots B(x, n)] A_n^{-1} \gamma)}{\sqrt{(\Sigma_0(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1}) - \gamma^T A_n^{-1} \gamma)}} \right]^2. \end{aligned}$$

We now compute $a_n(x)$,

$$\begin{aligned} a_n(x) &= \mathbb{E}[\mu_n(x, w)] \\ &= \mathbb{E}[\mu_0(x, w)] \\ &\quad + [B(x, 1) \cdots B(x, n)] A_n^{-1} \begin{pmatrix} y_1 - \mu_0(x_1, w) \\ \vdots \\ y_n - \mu_0(x_n, w) \end{pmatrix}. \end{aligned}$$

In the particular case that we use the squared exponential kernel, and w follows a normal distribution ($w_i \sim N(\mu_i, \sigma_i^2)$) and its components are independent, we have that

$$\begin{aligned} B(x, i) &= \int \Sigma_0(x, w, x_i, w_i) dw \\ &= \sigma_0^2 \exp\left(-\sum_{k=1}^n \alpha_{1,k} [x_k - x_{i,k}]^2\right) \prod_{k=1}^{d_1} \int \exp\left(-\alpha_{2,k} [w_k - w_{i,k}]^2\right) p(w_k) dw_k \end{aligned}$$

for $i = 1, \dots, n$.

We can also compute $\int \exp\left(-\alpha_{2,k} [w_k - w_{i,k}]^2\right) dp(w_k)$ for any k and i ,

$$\begin{aligned}
& \int \exp\left(-\alpha_{2,k} [w_k - w_{i,k}]^2\right) dp(w_k) \\
&= \frac{1}{\sqrt{2\pi}\sigma_k} \int \exp\left(-\alpha_{2,k} [z - w_{i,k}]^2 - \frac{[z - \mu_k]^2}{2\sigma_k^2}\right) dz \\
&= \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{\mu_k^2}{2\sigma_k^2} - \alpha_{2,k} w_{i,k}^2 - \frac{\left(\frac{\mu_k}{\sigma_k^2} + 2\alpha_{2,k} w_{i,k}\right)^2}{4\left(-\alpha_{2,k} - \frac{1}{2\sigma_k^2}\right)}\right) \\
&\quad \times \int \exp\left(-\left(\alpha_{2,k} + \frac{1}{2\sigma_k^2}\right) \left[z - \frac{\frac{\mu_k}{\sigma_k^2} + 2\alpha_{2,k} w_{i,k}}{2\left(b + \frac{1}{2\sigma_k^2}\right)}\right]^2\right) dz \\
&= \frac{1}{\sqrt{2}\sigma_k} \frac{1}{\sqrt{\alpha_{2,k} + \frac{1}{2\sigma_k^2}}} \exp\left(-\frac{\mu_k^2}{2\sigma_k^2} - \alpha_{2,k} w_{i,k}^2 - \frac{\left(\frac{\mu_k}{\sigma_k^2} + 2\alpha_{2,k} w_{i,k}\right)^2}{4\left(-\alpha_{2,k} - \frac{1}{2\sigma_k^2}\right)}\right).
\end{aligned}$$

Computation of the Value of Information and Its Gradient

Computation of the Value of Information

In this section, we prove the Lemma 1 of the paper.

Proposition 1. We have that

$$a_{n+1}(x) \mid \mathcal{F}_n, (x_{n+1}, w_{n+1}) \sim N(a_n(x), \tilde{\sigma}_n^2(x, x_{n+1}, w_{n+1}))$$

where

$$\tilde{\sigma}_n^2(x, x_{n+1}, w_{n+1}) = \text{Var}_n[G(x)] - \mathbb{E}_n[\text{Var}_{n+1}[G(x)] \mid x_{n+1}, w_{n+1}].$$

Proof. By equation 9,

$$a_{n+1}(x) = \mathbb{E}[\mu_{n+1}(x, w)] = \mathbb{E}[\mu_0(x, w)] + [B(1) \cdots B(n+1)] A_{n+1}^{-1} \begin{pmatrix} y_1 - \mu_0(x_1, w_1) \\ \vdots \\ y_{n+1} - \mu_0(x_{n+1}, w_{n+1}) \end{pmatrix}. \quad (9)$$

Since y_{n+1} conditioned on $\mathcal{F}_n, x_{n+1}, w_{n+1}$ is normally distributed, then $a_{n+1}(x) \mid \mathcal{F}_n, x_{n+1}, w_{n+1}$ is also normally distributed. By the tower property,

$$\begin{aligned}
\mathbb{E}_n[a_{n+1}(x) \mid x_{n+1}, w_{n+1}] &= \mathbb{E}_n[\mathbb{E}_{n+1}[G(x)] \mid x_{n+1}, w_{n+1}] \\
&= \mathbb{E}_n[G(x)] \\
&= a_n(x)
\end{aligned}$$

and

$$\begin{aligned}\tilde{\sigma}_n^2(x, x_{n+1}, w_{n+1}) &= \text{Var}_n[\mathbb{E}_{n+1}[G(x)] \mid x_{n+1}, w_{n+1}] \\ &= \text{Var}_n[G(x)] - \mathbb{E}_n[\text{Var}_{n+1}[G(x)] \mid x_{n+1}, w_{n+1}].\end{aligned}$$

This proves the proposition.

Proof of Lemma 1. Using the equation (9) and the previous proposition, we get the following formula for a_{n+1}

$$a_{n+1} = a_n + \tilde{\sigma}_n(x, x_{n+1}, w_{n+1})Z$$

where $Z \sim N(0, 1)$, which is the Lemma 1 of the paper.

Computation of the Gradient of the Value of Information

In this section, we compute the gradient of the value of information.

First, we compute the gradient in the general case,

$$\begin{aligned}\nabla V_n(x_{n+1}, w_{n+1}) &= \nabla h(a_n(A'), \tilde{\sigma}_n(A', x_{n+1}, w_{n+1})) \\ &= \sum_{i=1}^{l-1} (f_{j_{i+1}} - f_{j_i}) (-\Phi(-|c_i|)) \nabla(|c_i|) - (\nabla f_{j_{i+1}} - \nabla f_{j_i}) f(-|c_i|) \\ &= \sum_{i=1}^{l-1} (\nabla f_{j_{i+1}} - \nabla f_{j_i}) (-\Phi(-|c_i|) |c_i| - f(-|c_i|)) \\ &= \sum_{i=1}^{l-1} (-\nabla f_{j_{i+1}} + \nabla f_{j_i}) \varphi(|c_i|).\end{aligned}$$

We only need to compute ∇f_{j_i} for all i ,

$$\begin{aligned}\nabla \tilde{\sigma}_n(x, x_{n+1}, w_{n+1}) &= \nabla \left(\sqrt{(\text{Var}_n[G(x)] - \mathbb{E}_n[\text{Var}_{n+1}[G(x)] \mid x_{n+1}, w_{n+1}])} \right) \\ &= \beta_1 \left(\nabla B(x, n+1) - \nabla(\gamma^T) A_n^{-1} \begin{bmatrix} B(x, 1) \\ \vdots \\ B(x, n) \end{bmatrix} \right)\end{aligned}\tag{10}$$

$$-\frac{1}{2} \beta_1^3 \beta_2 [\nabla \Sigma_0(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1}) - 2 \nabla(\gamma^T) A_n^{-1} \gamma]\tag{11}$$

where

$$\begin{aligned}\beta_1 &= [\Sigma_0(x_{n+1}, w_{n+1}, x_{n+1}, w_{n+1}) - \gamma^T A_n^{-1} \gamma]^{-1/2} \\ \beta_2 &= B(x, n+1) - [B(x, 1) \cdots B(x, n)] A_n^{-1} \gamma.\end{aligned}$$

Now, we give a closed formula for this gradient when we use the squared exponential kernel, and w follows a normal distribution ($w_i \sim N(\mu_i, \sigma_i^2)$) and its components are independent. Observe that we can compute (10) explicitly by plugging in

$$\begin{aligned}\nabla_{x_{n+1}, j} \Sigma_0(x_{n+1}, w_{n+1}, x_i, w_i) &= \begin{cases} 0, & i = n+1 \\ -2\alpha_{1,j} [x_{n+1,j} - x_{i,j}] \Sigma_0(x_{n+1}, w_{n+1}, x_i, w_i), & i < n+1 \end{cases} \\ \nabla_{w_{n+1}, j} \Sigma_0(x_{n+1}, w_{n+1}, x_i, w_i) &= \begin{cases} 0, & i = n+1 \\ -2\alpha_{2,j} [w_{n+1,j} - w_{i,j}] \Sigma_0(x_{n+1}, w_{n+1}, x_i, w_i), & i < n+1 \end{cases}\end{aligned}$$

where $\nabla_{x_{n+1,j}}$ is the derivative respect to the j th entry of x_{n+1} . Finally, we only need to compute

$$\begin{aligned}\nabla_{x_{n+1,j}} B(x, n+1) &= -2\alpha_1^{(j)} (x_j - x_{n+1,j}) B(x, n+1) \\ \nabla_{w_{n+1,k}} B(x, n+1) &= \sigma_0^2 \exp\left(-\sum_{i=1}^n \alpha_1^{(i)} [x_i - x_{n+1,i}]^2\right) \prod_{j \neq k} \int \exp\left(-\alpha_2^{(j)} [w_j - w_{n+1,j}]^2\right) dp(w_j) \\ &\quad \times \int \left(-2\alpha_2^{(k)} (w_k - w_{n+1,k})\right) \exp\left(-\alpha_2^{(k)} [w_k - w_{n+1,k}]^2\right) dp(w_k).\end{aligned}$$

Acknowledgments

Peter Frazier and Saul Toscano-Palmerin were partially supported by NSF CAREER CMMI-1254298, NSF CMMI-1536895, NSF IIS-1247696, AFOSR FA9550-12-1-0200, AFOSR FA9550-15-1-0038, and AFOSR FA9550-16-1-0046.

References

- Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.
- Citi (2015). Citi bike website. <https://www.citibikenyc.com/>, accessed May 2015.
- Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons.
- Frazier, P., Powell, W., and Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, **21**(4), 599–613.
- Frazier, P. I. and Wang, J. (2015). Bayesian optimization for materials design. *arXiv 1506.01349*.
- Glasserman, P. (2003). *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media.
- Howard, R. (1966). Information Value Theory. *Systems Science and Cybernetics, IEEE Transactions on*, **2**(1), 22–26.
- Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006). Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, **34**(3), 441–466.
- Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, **13**(4), 455–492.
- Kraft, D. *et al.* (1988). *A software package for sequential quadratic programming*. DFVLR Obersaffeuhausen, Germany.
- Kushner, H. J. (1964). A new method of locating the maximum of an arbitrary multi-peak curve in the presence of noise. *Journal of Basic Engineering*, **86**, 97–106.
- Mockus, J. (1989). *Bayesian approach to global optimization: theory and applications*. Kluwer Academic, Dordrecht.

- Mockus, J., Tiesis, V., and Zilinskas, A. (1978). The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, **2**(117-129), 2.
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.
- O'Hagan, A. (1991). Bayes–hermite quadrature. *Journal of statistical planning and inference*, **29**(3), 245–260.
- Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.
- Scott, W., Frazier, P., and Powell, W. (2011). The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, **21**(3), 996–1026.
- Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.
- Villemonteix, J., Vazquez, E., and Walter, E. (2009). An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, **44**(4), 509–534.
- Xie, J., Frazier, P., Sankaran, S., Marsden, A., and Elmohamed, S. (2012). Optimization of computationally expensive simulations with gaussian processes and parameter uncertainty: Application to cardiovascular surgery. In *50th Annual Allerton Conference on Communication, Control, and Computing*.